Automatic primitive finding for action modeling

Baby, Sanmohan ; Krüger, Volker

Published in:
Proceedings of First International Workshop, Themis 2008

Publication date:
2008

Document Version
Publisher's PDF, also known as Version of record

Automatic primitive finding for action modeling

Sanmohan and Volker Krüger

Abstract

There has been a recent interest in segmenting action sequences into meaningful parts (action primitives) and to model actions on a higher level based on these action primitives. Unlike previous works where action primitives are defined a-priori and search is made for them later, we present a sequential and statistical learning algorithm for automatic detection of the action primitives and the action grammar based on these primitives. We model a set of actions using a single HMM whose structure is learned incrementally as we observe new types. Actions are modeled with sufficient number of Gaussians which would become the states of an HMM for an action. For different actions we find the states that are common in the actions which are then treated as an action primitive.

Keywords
Imitation learning, High-level behaviour recognition and scene understanding,Natural-language description of human behaviours, generative and discriminative models .

1 Introduction

Similar to phonemes being the building blocks of human language there is biological evidence that human action execution and understanding is also based on a set primitives [1]. But the notion of primitives for action does not only appear in neuro-biological papers. Also in the vision community, many authors have discussed that it makes sense to define a hierarchy of different action complexities such as *movements*, *activities* and *actions* [2]In terms of Bobick's notations, *movements* are action primitive, out of which activities and actions are composed.

Many authors use this kind of hierarchysee here the review by Moeslund et al[8]. One way to use such a hierarchy is to define a set of action primitives in connection with a stochastic grammar that uses the primitives as its alphabet. There are many advantages of using primitives are, for example: (1) The use of primitives and grammars is often more intuitive for the human which simplifies verification of the learning results by an expert. (2)*Parsing* primitives for recognition instead of using the signal directly leads to a better robustness under noise [9, 13]; (3) AI provides powerful techniques for higher level processing such as planning and plan recognition based on primitives and parsing. In some cases, it is reasonable to define the set of primitives and grammars by hand. In other cases, however, one would wish to compute the primitives and the stochastic grammar automatically based on a set of training observations. Examples for this can be found in robotics , surveillance, and DNA sequencing.

In this paper, we present an HMM-based approach to learn primitives and the corresponding stochastic grammar based on a set of training observations. Our approach is able to learn on-line and is able to refine the representation when newly incoming data

supports it. We test our approach on a typical surveillance scenario similar to [11] and on the data used in [13] for human arm movements.

A number of authors represent action in a hierarchical manner. Staffer and Grimson [11] compute for a surveillance scenario a set of action primitives based on co-occurrences of observations. This work is used to motivate the surveillance setup of one of our experiments.

In [10] Robertson and Reid present a full surveillance system that allows high-level behavior recognition based on simple actions. Their system seems to require human interaction in the definition of the primitive actions such as *walking, running, standing, dithering* and the qualitative positions (*nearside-pavement, road, driveway, etc*). This is, what we would like to automate.

In [3] actions are recognized by computing the cost through states an action pass through. The states are found by k-means clustering on the prototype curve that best fits sample points according to a least square criterion. Hong et al [7] built a Finite State Machine for recognition by building individual FSM s for each gesture. Fod et al. [4] uses a segmentation approach using zero velocity crossing. Primitives are then found by clustering in the projected space using PCA. In these methods the k-means clustering can cause the points belong to states or segments violating temporal order. Our approach overcomes this problem very efficiently.

The idea of segmenting actions into atomic parts and then modeling the temporal order using Stochastic Context Free Grammar is found in [6]. In [5], signs of first and second derivatives are used to segment action sequences. These works require the storage of all training data if one wishes to modify the model to accommodate a new action. Our approach eliminate this requirement and thus make it suitable for imitation learning.

Our idea of merging of several HMMs to get a more complex and general model is found in [12]. The merging strategy explained in [12] is applicable for discrete HMMs only. We introduce a way of merging for the continuous case. New models can be introduced and merged online.

1.1 Problem Statement

We define two sets of primitives. One set contains parts that are unique to one *type* of action and another set that contains parts that are common to more than one *type* of action. Two sequences are of the same *type* if they do not differ significantly, e.g., two different walking paths. Hence we attempt to segment sequences into parts that are not shared and parts that are common across sequences types. Then each sequence will be a combination of these segments. We also want to generate rules that govern the interaction among the primitives. Keeping this in mind we state our objectives as:

1. Let $\mathscr{L} = \{X_1, X_2, \cdots, X_m\}$ be a set of data sequences where each X_i is of the form $x_1^i x_2^i \cdots, x_{T_i}^i$ and $x_j^i \in \mathbb{R}^n$. Let these observations be generated from a finite set of sources (or states) $\mathscr{S} = \{s_1, s_2, \cdots s_r\}$. Let $S_i = s_1^i s_2^i \cdots, s_{T_i}^i$ be the state sequence associated with X_i. Find a partition $\mathscr{S}'$ of the set of states $\mathscr{S}$ where $\mathscr{S}' = \mathscr{A} \cup \mathscr{B}$ such that $\mathscr{A} = \{a_1, a_2, \cdots, a_k\}$ and $\mathscr{B} = \{b_1, b_2, \cdots, b_l\}$ are sets of state subsequences of X_i's and each of the a_i's appear in more than one state sequence and each of the b_j's appear in exactly one of the state sequence. The set $\mathscr{A}$ corresponds to common actions and the set $\mathscr{B}$ correspond to unique parts.

2. Generate a grammar with elements of $\mathscr{S}'$ as symbols which will generate primitive sequences that match with the data sequences.

2 Modeling the Observation Sequences.

We take the first sequence of observations X_1 with data points $x_1^1 x_2^1 \cdots x_{T_1}^1$ and generate a few more spurious sequences of the same type by adding Gaussian noise to it. Then we choose $(\mu_i^1, \sigma_i^1),\ i = 1,2,...k^1$ so that parts of the data sequence are from $\mathscr{N}(\mu_i^1, \Sigma_i^1)$ in that order. The value of k^1 is such that $\mathscr{N}(\mu_i^1, \Sigma_i^1),\ i = 1,2,...k^1$ will cover the whole data. This value is not chosen before hand and varies with the variation and length of the data.

The next step is to make an HMM $\lambda_1 = (A^1, B^1, \pi^1)$ with k^1 states . We let A^1 to be a left-right transition matrix and $B_j^1(x) = \mathscr{N}(x, \mu_j^1, \Sigma_j^1)$. All the states at this stage get a label 1 to indicate that they are part of sequence type 1. This model will now be modified recurcively.

Now we will modify this model by adding new states to it or by modifying the current output probabilities of states so that the modified model λ_M will be able to generate new types of data with high probability.

Let $n-1$ be the number of types of data sequences we have seen so far. Let X_c be the next data sequence to be processed. Calculate $P(X_c|\lambda_M)$ where λ_M is the current model at hand. A low value for $P(X_c|\lambda_M)$ indicates that the current model is not good enough to model the data sequences of type X_c and hence we make a new HMM λ_c for X_c as described in the beginning and the states are labeled n. The newly constructed HMM λ_c will be merged to λ_M so that the updated λ_M will be able to generate data sequences of type X_c.

Suppose we want to merge λ_c into λ_M so that $P(X_k|\lambda_M)$ is high if $P(X_k|\lambda_c)$ is high. Let $C_c = \{s_{c1}, s_{c2}, \cdots, s_{ck}\}$ and $C_M = \{s_{M1}, s_{M2}, \cdots, s_{Ml}\}$ be the set of states of λ_c and λ_M respectively. Then the state set of the modified λ_M will be $C_M \cup D_1$ where $D_1 \subseteq C_c$. Each of the states s_{ci} in λ_c affects λ_M in one of the following ways:

1. If $d(s_{ci}, s_{Mj}) < \theta$, for some $p \in \{1,2,\cdots l\}$, then s_{ci} and s_{Mj} will be merged into a single state. Here d is a distance measure and θ is a threshold value. The output probability distribution associated with s_{Mj} is modified to be a combination of the existing distribution and ${b^k}_{s_{ci}}(x)$. Thus ${b^M}_{Mj}(x)$ is a mixture of Gaussians. We append n to the label of the state s_{Mj}. All transitions to s_{ci} are redirected to s_{Mj} and all transitions from s_{ci} will now be from s_{Mj}. The basic idea behind merging is that we do not need two different states which describe the same part of the data.

2. If $d(s_{ci}, s_{Mj}) > \theta,\ \forall j$, a new state is added to λ_M. i.e. $s_{ci} \in D_1$. Let s_{ci} be the r^{th} state to be added from λ_c. Then, s_{ci} will become the $(Ml+r)^{th}$ state of λ_M. The output probability distribution associated with this new state in λ_M will be the same as it was in λ_c. Hence ${b^M}_{Ml+r}(x) = \mathscr{N}(x, \mu_{s_{c_i}}, \Sigma_{s_{c_i}})$. Initial and transition probabilities of λ_M are adjusted to accommodate this new state. The newly added state will keep its label n.

We use Kullback-Leibler Divergence to calculate the distance between states. The

K-L divergence from $\mathcal{N}(x,\mu_0,\Sigma_0)$ to $\mathcal{N}(x,\mu_1,\Sigma_1)$ has a closed form solution given by :

$$D_{KL}(Q||P) = \frac{1}{2}\left(\log\frac{|\Sigma_1|}{|\Sigma_0|} + \mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\mu_1-\mu_0)^T\Sigma_1^{-1}(\mu_1-\mu_0) - n\right) \quad (1)$$

Here n is the dimension of the space spanned by the random variable x.

Now we elaborate more on the addition and merging of states into the combined model. Our aim is to make the new model compatible with the newly observed type of data sequences. Since the states are probability distributions, if we see that two probability distributions corresponding to different states are very close we do not need to keep them apart. Keeping these two states together will help us to model the observations generated from two distributions by a single one. We use (1) to compute the similarity measure of two states. We can observe that (1) will not handle mixture of Gaussians. We still use this equation to evaluate componetwise distances in mixtures and check if any of the components are close to the distribution we are testing. We justify this criteria since our aim is to find out if a new state is to be embedded into another state or not.

2.1 Finding Primitives

When all sequences have been processed, we apply Viterbi algorithm on the final merged model λ_M, and find the hidden states associated with each of the sequences. Let $P_1, P_2, \cdots P_r$ be different Viterbi paths at this stage. Since we want the common states that are contiguous across state sequences, it is similar to finding the longest common substring(LCS) problem. We take all paths with non-empty intersection and find the largest common substring a_k for them. Then a_k is added to $\mathscr{A}$ and is replaced with an empty string in all the occurrences of a_k in P_i, $i = 1,2,\cdots r$.

We continue to look for largest common substings until we get an empty string as the common substring for any two paths. Thus we end up with new paths $P_1', P_2', \cdots P_r'$ where each P_i' consists of one or more segments with empty string as the separator[1]. These remaining segments in each P_i' are unique to P_i. Each of them are also primitives and form the members of the set $\mathscr{B}$. Our objective was to find these two sets $\mathscr{A}$ and $\mathscr{B}$ as was stated in Sec. 1.1.

3 Generating the grammar for primitives

Let $\mathscr{S}' = \{c_1, c_2, \cdots c_p\}$ be the set of primitives available to us. We wish to generate rules of the form $P(c_i \rightarrow c_j)$ which will give the likelihood of occurrence of the primitive c_j followed by primitive c_i. We do this by constructing a directed graph G which encodes the relations between the primitives. Using G we will derive a formal grammar for the elements in $\mathscr{S}'$.

Let n be the number of types of data that we have processed. Then each of the states in our final HMM λ_M will have labels from a subset of $\{1,2,\cdots,n\}$, see Fig.1. By way of definition each of the states that belong to a primitive c_i will have the same label set l^{c_i}. Let $\mathscr{L} = \{l_1, l_2 \cdots, l_p\}$ $p \geq n$ be the set of different type of labels received by the primitives.Let $G = (V,E)$ be a directed graph where $V = \mathscr{S}'$ and $e_{ij} = (c_i, c_j) \in E$ if there

[1]The segmentation is caused by the gaps produced by the removal of elements of $\mathscr{A}$.

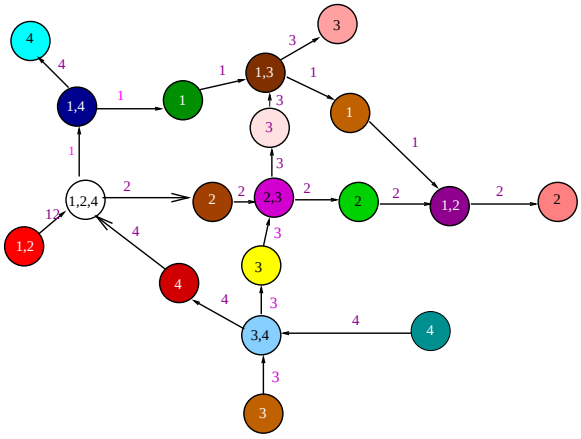

Figure 1: Directed graph for finding the grammar

is a path $P_k = \cdots c_i c_j \cdots$ for some k. We have given the directed graph constructed for out test data in Fig. 1.

We proceed to derive a precise Stochastic Context Free Grammar (SCFG) from the directed graph G we have constructed. Let $N = \mathscr{S}'$ be the set of terminals. To each vertex c_i with an outgoing edge with label $l^{e_{ij}}$, associate a corresponding non-terminal $A_{c_i}^{l^{e_{ij}}}$. Let $\mathscr{N} = S \cup \{A_{c_i}^{l^{e_{ij}}}\}$ be the set of all non-terminals where S is the start symbol. For each primitive c_i that occurs at the start of a sequence and connecting to c_j define the rule

$$S \longrightarrow c_i A_{c_j}^{l^{c_i}} \tag{2}$$

To each of the internal nodes c_j with an incoming edge e_{ij} connecting from c_i and an outgoing edge e_{jk} connecting to c_k define the rule

$$A_{c_i}^{l^{c_i} \cap l^{c_j}} \longrightarrow c_j A_{c_k}^{l^{c_j} \cap l^{c_k}} \tag{3}$$

For each leaf node c_j with an incoming edge e_{ij} connecting from c_i and no outgoing edge define the rule

$$A_{c_j}^{l^{c_i} \cap l^{c_j}} \longrightarrow \varepsilon \tag{4}$$

The symbol ε denotes an empty string. We assign equal probabilities to each of the expansions of a nonterminal symbol except for the expansion to an empty string which occurs with probability 1. Thus

$$P(A_{c_i}^{l_{ij}} \longrightarrow c_j A_{c_j}^{l_{jk}}) = \frac{1}{|c_i^{(o)}|} \text{ if } |c_i^{(o)}| > 0 \tag{5}$$

$$P(A_{c_i}^{l^{e_{ij}}} \longrightarrow \varepsilon) = 1 \text{ if } |c_i^{(o)}| = 0 \tag{6}$$

where $|c_i^{(o)}|$ represents the number of outgoing edges from c_i and $l_{mn} = l^{c_m} \cap l^{c_n}$. Let $\mathscr{R}$ be the collection of all rules given in (2), (3), and (4). For each $r \in \mathscr{R}$ associate a probability

$P(r)$ using (5) and (6). Then $(\mathscr{N}, \mathscr{S}', S, \mathscr{R}, P(.))$ is the stochastic grammar that models our primitives.

One might wonder why the HMM λ_M is not enough to describe the grammatical structure of the observations and why the SCFG is necessary. The HMM λ_M would have been sufficient for a single observation type. However for several observation types as in final λ_M, regular grammars, as modeled by HMMs are usually too limited to model the different observation types so that different observation types can be confused.

4 Experiments

We have run three experiments: In the first experiment we generate a simple data set with very simple cross-shaped paths. The second experiment is motivated by the surveillance scenario of Stauffer and Grimson [11] and shows a complex set of paths as found outside our building. The third experiment is motivated by the work of Vincente and Kragic [13] on the recognition of human arm movements.

4.1 Testing on Simulated Data

We illustrate the result of testing our method on a set of two sequences generated with mouse clicks. The original data set for testing is shown in Fig. 1 We have two paths one from A to B and the other from C to D. These paths intersect in the middle at E . If we were to remove the points around E we will the have segments AE, EB, CE and ED. We extracted these segments with the above mentioned procedure. When the model merging took place, the overlapping states in the middle were merged into one. The result is shown in Fig. 2. The primitives that we get are colored. As one can see in Fig. 2, primitive b is a common primitive and belongs to our set A, primitives a,c,d,e belong to our set B.

4.2 2D-Trajectory Data

The second experiment was done on a surveillance-type data inspired by [11]. The paths represent typical walking paths outside of our building. In this data there are four different types of trajectories with heavy overlap, see Fig. 2 bottom left. We can also observe that the data is quite noisy.

The result of primitive segmentation is shown in Fig. 2 on the bottom right. Different primitives are colored differently and we have named the primitives with different letters. As one can see, our approach results in primitives that coincide roughly with our intuition. Furthermore, our approach is very robust even with such noisy observations and lot of overlaps.

4.3 Hand gesture data

Finally, we have tested our approach on the dataset provided by Vincente and Kragic [13]. In that data, several volunteers performed a set of simple arm movements such as *grasp object*, *rotate object move object by lifting*,*push object aside* and *push object forward*. Five different actions are considered: a) pick up an object from a table, b) rotate an object on a table, c) push an object forward, d) push an object to the side, and e) move an object to the side by picking it up. Each action is performed in 12 different conditions: two

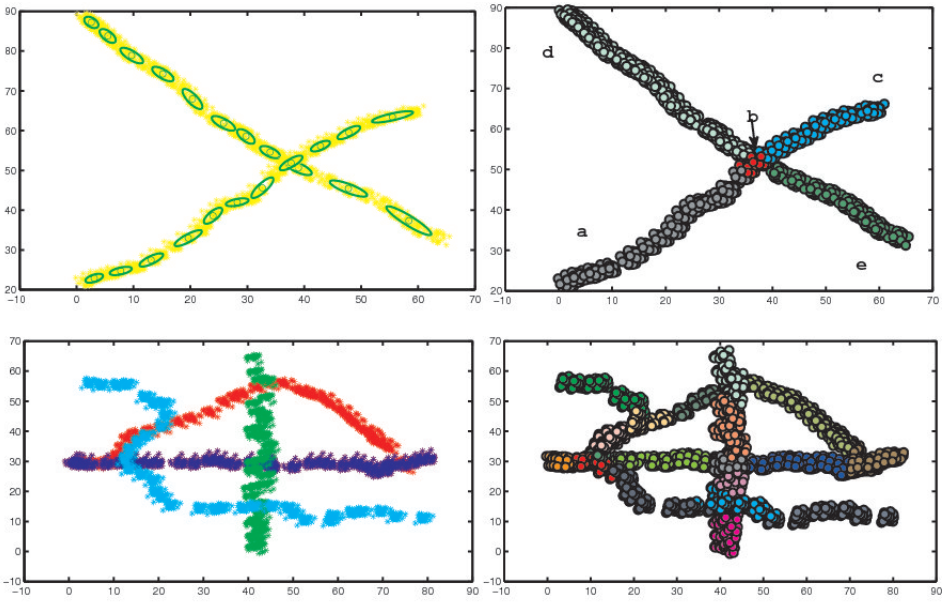

Figure 2: The top left figure shows the simulated 2d data sequences. The top right figure shows the finally detected primitives with different colors. Primitive b is a common primitive and belongs to set $\mathscr{A}$, primitives a,c,d,e belong to set $\mathscr{B}$. The bottom left figure shows trajectories from tracking data. Each type is colored differently. Only a part of the whole data is shown. The bottom right figure shows the detected primitives. Each primitive is colored differently.

different heights, two different locations on the table, and having the demonstrator stand in three different locations (0, 30, 60 degrees) . Furthermore, all actions are demonstrated by 10 different people. The movement is measured using magnetic sensors placed on: chest, back of hand, thumb, and index finger. The observation in [13] was that for many arm movements the approach movement towards an object was followed by a grasping of the object so that they could naturally be decoupled. By considering two different models for manually segmented primitives they found that the recognition rate was much higher when the primitives that are common across actions are considered separately. Using their dataset, our approach is able to provide the primitives and the grammar automatically. We have used the 3-d coordinates from the sensors along with velocity of the hand. The forward and backward motion of the hand trajectory belongs to different states even though they spatially adjacent. The temporal order of primitives for actions for a particular position are shown in 3(a). We have used the Natural Language Toolkit (NLTK, *http://nltk.sourceforge.net*) for parsing. The parsing tree for *push-object forward* action is shown in Fig. 3(b)

All these actions started from rest pose and ended at almost same position. We were able to extract these obvious common primitives accurately. The large number of prim-

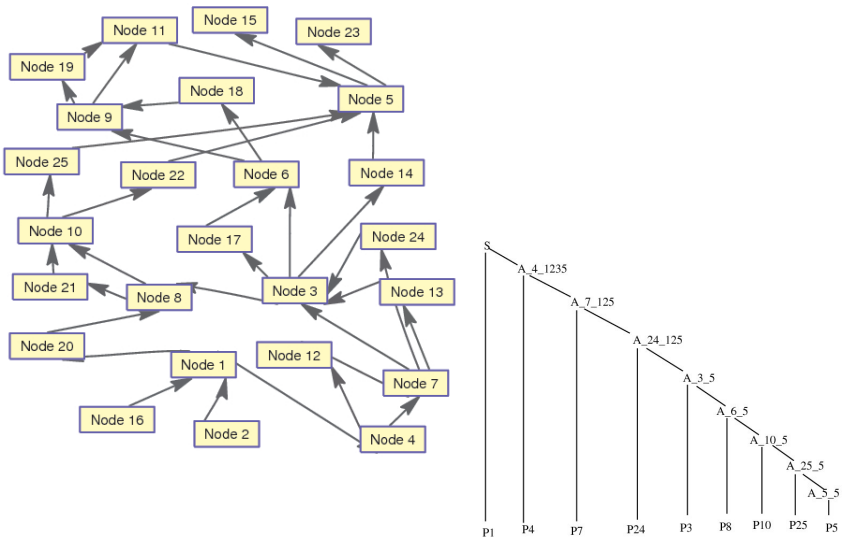

Figure 3: Left figure shows the temporal order for primitives of hand gesture data. Node number corresponds to different primitives. Right figure shows the parse tree from NLTK for *push-object forward* action.

itives are due to the variations in repetitions by several subjects. We have experimented with supervised placing of states along tranjectories and ended up with less number of primitives. But the unsupervised learning is more challenging and interesting and hence we ignore the number of primitives we end up with.

5 Conclusions

We have presented and tested an approach for automatically computing a set of primitives and the corresponding stochastic context free grammar from a set of training observations. Our stochastic regular grammar is closely related to the usual HMMs. One important difference between common HMMs and a stochastic grammar with primitives is that with usual HMMs, each trajectory (action, arm movement, etc.) has its own, distinct HMM. This means that the set of HMMs for the given trajectories are not able to reveal any commonalities between them. In case of our arm movements, this means that one is not able to deduce that some actions share the grasp movement. Using the primitives and the grammar, this is different. Here, the different actions share common primitives which would even allow to use AI techniques for, e.g., planning or plan recognition. Another important aspect of our approach is that we can modify our model to include a new action without requiring the storage of previous actions for it.

Many authors point at the huge task of learning parameters and the size of training data for an HMM when the number of states are increasing. But in our method, transition, initial and observation probabilities for all states are assigned during our merging phase and hence the use of EM algorithm is not required. Thus our method is scalable to the

number of states. Our approach of using states have a close connection to [3] but our method is superior in preserving the temporal order and hence in recognition.

It is interesting to note that stochastic grammars are closely related to Belief networks where the hierarchical structure coincides with the production rules of the grammar. We will further investigate this relation ship in future work.

In future work, we will also evaluate the performance of normal and abnormal path detection using our primitives and grammars.

References

[1] Giszter SF Loeb E Mussa-Ivaldi FA Saltiel P. Bizzi E. Modular organization of motor behavior in the frog's spinal cord. *Trends Neurosci.*, 18(10):442–446, 1995.

[2] A. Bobick. Movement, Activity, and Action: The Role of Knowledge in the Perception of Motion. *Philosophical Trans. Royal Soc. London*, 352:1257–1265, 1997.

[3] A.F. Bobick and A.D. Wilson. A state-based approach to the representation and recognition of gesture. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(12):1325–1337, Dec 1997.

[4] Ajo Fod, Maja J. Matarić, and Odest Chadwicke Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 12(1):39–54, 2002.

[5] G. Guerra-Filho and Y. Aloimonos. A sensory-motor language for human activity understanding. *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 69–75, 4-6 Dec. 2006.

[6] Cornelia Fermüller Gutemberg Guerra-Filho and Yiannis Aloimonos. Discovering a language for human activity. In *AAAI 2005 Fall Symposium on Anticipatory Cognitive Embodied Systems, Washington, D.C*, pages 70–77, 2005.

[7] P. Hong, M. Turk, and T. Huang. Gesture modeling and recognition using finite state machines, 2000.

[8] T. Moeslund, A. Hilton, and V. Krueger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–127, 2006.

[9] L.R. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, 1993.

[10] N. Robertson and I. Reid. Behaviour Understanding in Video: A Combined Method. In *Internatinal Conference on Computer Vision*, Beijing, China, Oct 15-21, 2005.

[11] C. Stauffer and W.E.L. Grimson. Learning Patterns of Activity Using Real-Time Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[12] A. Stolcke and S. M. Omohundro. Best-first model merging for hidden Markov model induction. Technical Report TR-94-003, 1947 Center Street, Berkeley, CA, 1994.

[13] I. S. Vicente, V. Kyrki, and D. Kragic. Action recognition and understanding through motor primitives. *Advanced Robotics*, 21:1687–1707, 2007.